

SYNTHESIS OF MINIMAL ERROR CONTROL SOFTWARE

RUPAK MAJUMDAR¹, INDRANIL SAHA², AND MAJID ZAMANI³

ABSTRACT. Software implementations of controllers for physical systems are at the core of many embedded systems. The design of controllers uses the theory of dynamical systems to construct a mathematical control law that ensures that the controlled system has certain properties, such as asymptotic convergence to an equilibrium point, while optimizing some performance criteria. However, owing to quantization errors arising from the use of fixed-point arithmetic, the implementation of this control law can only guarantee *practical* stability: under the actions of the implementation, the trajectories of the controlled system converge to a bounded set around the equilibrium point, and the size of the bounded set is proportional to the error in the implementation. The problem of verifying whether a controller implementation achieves practical stability for a given bounded set has been studied before. In this paper, we change the emphasis from verification to automatic *synthesis*. Using synthesis, the need for formal verification can be considerably reduced thereby reducing the design time as well as design cost of embedded control software.

We give a methodology and a tool to synthesize embedded control software that is Pareto optimal w.r.t. both performance criteria and practical stability regions. Our technique is a combination of static analysis to estimate quantization errors for specific controller implementations and stochastic local search over the space of possible controllers using particle swarm optimization. The effectiveness of our technique is illustrated using examples of various standard control systems: in most examples, we achieve controllers with close LQR-LQG performance but with implementation errors, hence regions of practical stability, several times as small.

1. INTRODUCTION

Software implementations of controllers for physical systems are the core of many critical cyber-physical systems. The design of these systems usually proceeds in two steps. First, starting with a mathematical model of the system, one designs a mathematical control law that ensures that the physical system, equipped with this control law, has certain desirable properties such as asymptotic stability (convergence to an ideal behavior) and performance. Second, the control law is implemented as a software task on a specific hardware architecture. Since the implementation has quantization errors due to the use of fixed-precision representation of real numbers, the quantization of a stabilizing controller may lead to limit cycles and chaotic behavior [Kal56]. Hence, the implemented system usually guarantees the weaker property of *practical* stability, where the system is guaranteed to converge to a bounded set around the ideal behavior and the size of the bounded set is proportional to the quantization error.

Much recent research has focused on verifying that a given implementation of a control law guarantees that the practical stability region lies within a given set [PW06, PW07, Fer10, AMST10, DM11]. In this paper, we change the emphasis from verification to *synthesis*. We provide a design methodology to synthesize a control implementation for which the effect of implementation errors on system performance is minimized. Hence, the need for verification can be substantially reduced.

We focus on linear systems in this paper. For linear systems, a standard optimal control design approach uses the *linear quadratic regulator* (LQR) and *linear quadratic Gaussian* (LQG) algorithms [Hes09], which finds a feedback controller stabilizing the plant while minimizing quadratic cost functions. The LQR cost function takes into account the deviations of the state and control inputs from ideal values and the LQG cost function takes into account the deviation of the state from its estimation. However, in general, they do not take implementation errors arising from fixed-precision arithmetic into account. Thus, a controller optimizing only the LQR-LQG cost may have a large implementation error because its implementation on a fixed-precision

platform has large numerical errors, but a controller “close” to the optimal performance may have much lower numerical errors when implemented on the same platform.

In our methodology, we modify the LQR-LQG performance criterion to additionally minimize the error due to quantization in the implementation. Technically, our methodology has two parts. First, how can we estimate the quantization error of a given implementation? Second, how can we find Pareto-optimal points for the two objectives given by the LQR-LQG and quantization error cost functions? We proceed as follows.

For the first step, for a given linear feedback controller and the operating intervals of the states of the plant and the controller, we first perform a precise range analysis of the controller variables, and use the computed ranges to allocate bitwidths to each controller variable. We implement our range analysis based on linear programming. Using the allocated bitwidths, we generate code for a fixed-precision program implementing the control law. Finally, we use an algorithm based on mixed-integer linear programming to find a bound on the maximum difference between the ideal control law and the output of the fixed-precision program.

For the second step, we optimize a weighted linear combination of the two cost functions using a stochastic local search technique. LQR-LQG is attractive because it gives rise to a *convex* optimization problem, for which efficient solutions are known. Unfortunately, additionally tracking the quantization error results in a non-convex optimization problem. We solve the non-convex optimization problem using *particle swarm optimization* (PSO), a population-based stochastic optimization approach [KE95, LAS09, JLY07]. PSO iteratively solves an optimization problem by maintaining a population (or *swarm*) of candidate controllers, called *particles*, and moving them around in the search-space of possible controllers, trying to minimize the objective function.

In more detail, our algorithm proceeds as follows. Given a linear control design problem, we set up a non-convex optimization problem to minimize a weighted combination of the LQR-LQG cost function and the implementation error. We minimize this cost function using PSO. In each step of PSO, given a new position of a particle, we check if the position represents a stabilizing controller (by examining the eigenvalues of the controlled system). If not, we assign the position an infinite cost. If the position represents a stabilizing controller, we generate the best possible fixed-point code for this controller under a hardware budget and perform static analysis to estimate a bound on the implementation error. We compute the value of the objective function by taking the weighted sum of the LQR-LQG cost and this bound. We continue PSO until convergence or until some iteration bound is met. At this point, we output the controller that minimized the objective function.

We have implemented this methodology on top of Matlab’s Control Theory Toolbox, using an implementation of PSO proposed in [EKG12], and a custom static analysis using the `lp_solve` linear programming tool. In our experiments, we compare the LQR-LQG cost and implementation errors of controllers generated by conventional LQR-LQG optimization (implemented in Matlab) with controllers generated by PSO using our methodology. In most cases, our controllers have LQR-LQG costs close to the optimal LQR-LQG controllers, but have implementation errors that are reduced by a factor of 4 or more. Thus, we generate controllers with guaranteed bounds on practical stability regions that are 4 times or more smaller than the pure LQR-LQG controllers. Our work provides, for the first time, an integrated analysis and tool to take quantization errors into account in model-based design and implementation of controllers.

Other Related Work Besides the related work mentioned above, we mention the results in [Wil85, Wil89, LSG92] which provide controller synthesis approaches minimizing some performance criteria while controllers are implemented using fixed-point arithmetic. The results in [Wil85, Wil89, LSG92] assume some excitation conditions under which the quantization error can be modeled as a zero mean uniform white noise. Furthermore, they do not provide any bounds on regions of practical stability. However, the result in this paper does not have any assumption on the quantization error and it provides an explicit bound on the regions of practical stability.

The range analysis problem has been studied extensively in the context of optimum bitwidth allocation to intermediate variables in a fixed-point program, mostly in the DSP domain. Both static [LGC⁺06, LCNT07, OCC⁺07] and simulation-based [BR05, MSBZ07] approaches have been used. Static approaches usually employ abstractions based on interval arithmetic [Moo66] or affine arithmetic [SF97]. Simulation-based methods, especially those performing constrained-random simulations, suffer from the lack of completeness. This causes the resultant system to be non-robust, incomplete simulation can lead to overflow conditions resulting in incorrect behavior. We found our mixed-integer linear programming approach to be both precise and reasonably fast for our application.

2. PRELIMINARIES

2.1. Controllers and Observers. We use symbols \mathbb{N}_0 , \mathbb{R} , and \mathbb{R}_0^+ for the set of nonnegative integers, real and nonnegative real numbers. For a vector $x \in \mathbb{R}^n$, we denote by x_i the i -th element of x , and by $\|x\|$ the Euclidean norm of x . Recall that $\|x\| = \sqrt{x_1^2 + x_2^2 + \dots + x_n^2}$. The symbols I_n and $0_{n \times m}$ denote the identity and zero matrices in $\mathbb{R}^{n \times n}$ and $\mathbb{R}^{n \times m}$, respectively. A continuous function $\gamma : \mathbb{R}_0^+ \rightarrow \mathbb{R}_0^+$, is said to belong to class \mathcal{K} if it is strictly increasing and $\gamma(0) = 0$; γ is said to belong to class \mathcal{K}_∞ if $\gamma \in \mathcal{K}$ and $\gamma(r) \rightarrow \infty$ as $r \rightarrow \infty$. A continuous function $\beta : \mathbb{R}_0^+ \times \mathbb{R}_0^+ \rightarrow \mathbb{R}_0^+$ is said to belong to class \mathcal{KL} if, for each fixed s , the map $\beta(r, s)$ belongs to class \mathcal{K}_∞ with respect to r and, for each fixed nonzero r , the map $\beta(r, s)$ is decreasing with respect to s and $\beta(r, s) \rightarrow 0$ as $s \rightarrow \infty$.

In this paper, we focus on *linear* control systems given by the differential equation:

$$(2.1) \quad \begin{cases} \dot{\xi} = A\xi + Bv + \bar{B}\omega, \\ \eta = C\xi + \nu, \end{cases}$$

where, for any $t \in \mathbb{R}$, $\xi(t) \in \mathbb{R}^n$, $v(t) \in \mathbb{R}^m$, $\omega(t) \in \mathbb{R}^q$, $\eta(t) \in \mathbb{R}^p$, and A , B , \bar{B} , and C are matrices of appropriate dimensions. The curve $\xi : \mathbb{R} \rightarrow \mathbb{R}^n$ is a *trajectory* of (2.1) if there exist curves $v : \mathbb{R} \rightarrow \mathbb{R}^m$ and $\omega : \mathbb{R} \rightarrow \mathbb{R}^q$ such that the time derivative of ξ satisfies (2.1). In the rest of the paper, we assume that all curves v and ω have some regularity assumptions, guaranteeing existence and uniqueness of the solutions of (2.1). Note that v , ω , η , and ν denote control input, disturbance, output of the system and measurement noise, respectively. We assume that $\omega(t)$ and $\nu(t)$, for any $t \in \mathbb{R}$, are zero-mean Gaussian noise processes (uncorrelated from each other). For all curves ω , we also write $\xi_{xv}(t)$ to denote the points reached at time t under the input v from initial condition $x = \xi_{xv}(0)$.

To describe the mismatch between the controller specifications and its software implementations such as digital sampling and finite precision arithmetic, which is the focus of this paper, we consider the discrete-time version of (2.1), as follows:

$$(2.2) \quad \begin{cases} x[r+1] = A_\tau x[r] + B_\tau u[r] + \bar{B}_\tau d[r] + e_s, \\ y[r] = Cx[r] + v[r], \end{cases}$$

where the matrices A_τ , B_τ , and \bar{B}_τ are given by:

$$A_\tau = e^{A\tau}, \quad B_\tau = \int_{r\tau}^{(r+1)\tau} e^{A(\tau-t)} B dt, \quad \bar{B}_\tau = \int_{r\tau}^{(r+1)\tau} e^{A(\tau-t)} \bar{B} dt,$$

and τ is the sampling time. The function e^{At} , for any $t \in \mathbb{R}$, denotes the matrix function defined by the convergent series:

$$e^{At} = I_n + At + \frac{1}{2!} A^2 t^2 + \frac{1}{3!} A^3 t^3 + \dots,$$

where e is Euler's constant. The signals x , u , d , y , and v describe the exact value of the signals ξ , v , ω , η , and ν , respectively, at the sampling instants $0, \tau, 2\tau, 3\tau, \dots$. Mathematically, we have:

$$x[r] = \xi(r\tau), \quad u[r] = v(r\tau), \quad d[r] = \omega(r\tau), \quad y[r] = \eta(r\tau), \quad v[r] = \nu(r\tau), \quad \forall r \in \mathbb{N}_0.$$

The term e_s in (2.2) is the sampling error. It can be shown that by sampling sufficiently fast, the error e_s can be made arbitrarily small [CF95]. Since typical embedded controller implementations use sampling time

in the range of milliseconds to microseconds, we will make the assumption that quantization errors dominate the sampling errors, and assume that $e_s = 0$.

We assume that only output of the system y is measurable and not the full state x . Hence, a (proportional) feedback $K : \mathbb{R}^n \rightarrow \mathbb{R}^m$ defines the input $u[r] = -K\hat{x}[r]$ based on an estimation \hat{x} of the state x . As explained in [Hes09], the estimation \hat{x} can be constructed using the observer dynamic:

$$(2.3) \quad \begin{cases} \hat{x}[r+1] = A_\tau \hat{x}[r] + B_\tau u[r] + L(y[r] - \hat{y}[r]), \\ \hat{y}[r] = C\hat{x}[r], \end{cases}$$

where \hat{y} should be viewed as an estimate of y and the linear map $L : \mathbb{R}^p \rightarrow \mathbb{R}^n$ is called an observer gain. By applying the feedback $u[r] = -K\hat{x}[r]$ and combining the dynamics of control system in (2.2) and observer in (2.3), one obtains:

$$(2.4) \quad \begin{cases} x[r+1] = A_\tau x[r] - B_\tau K\hat{x}[r] + \bar{B}_\tau d[r], \\ \hat{x}[r+1] = (A_\tau - B_\tau K - LC)\hat{x}[r] + LCx[r] + Lv[r]. \end{cases}$$

As shown in [AMST10], using a fixed-point implementation of the feedback gain as well as the observer dynamic, one gets the following overall dynamic:

$$(2.5) \quad \begin{cases} x[r+1] = A_\tau x[r] - B_\tau K\hat{x}[r] + \bar{B}_\tau d[r] + B_\tau e_{q2}, \\ \hat{x}[r+1] = (A_\tau - B_\tau K - LC)\hat{x}[r] + LCx[r] + Lv[r] + e_{q1}, \end{cases}$$

where e_{q1} and e_{q2} are quantization errors in observer dynamic and feedback gain codes, respectively. Now, one can rewrite the control system in (2.5) as follows:

$$(2.6) \quad w[r+1] = Gw[r] + H_1 e_1[r] + H_2 e_2[r],$$

with:

$$w = \begin{bmatrix} x \\ \hat{x} \end{bmatrix}, \quad e_1 = \begin{bmatrix} d \\ v \end{bmatrix}, \quad e_2 = \begin{bmatrix} e_{q1} \\ e_{q2} \end{bmatrix},$$

and:

$$G = \begin{bmatrix} A_\tau & -B_\tau K \\ LC & A_\tau - B_\tau K - LC \end{bmatrix}, \quad H_1 = \begin{bmatrix} \bar{B}_\tau & 0_{n \times p} \\ 0_{n \times q} & L \end{bmatrix}, \quad H_2 = \begin{bmatrix} 0_{n \times n} & B_\tau \\ I_n & 0_{n \times m} \end{bmatrix}.$$

Since states of the control system (2.1) are bounded physical quantities, such as temperature, pressure, position, velocity, acceleration and so on, their estimations and the output of the control system are bounded quantities as well. Hence, in the rest of the paper and without loss of generality, we assume that $y \in Y$, and $\hat{x} \in \hat{X}$, where $Y \subset \mathbb{R}^p$, and $\hat{X} \subset \mathbb{R}^n$ are compact.

2.2. Stability of perturbed systems. Here, we recall the notion of uniform global asymptotic stability with respect to a set, presented in [LSW96].

Definition 2.1 ([LSW96]). *A control system of the form (2.1) is uniformly globally asymptotically stable (UGAS) with respect to a set \mathcal{A} if there exists a \mathcal{KL} function β such that for any $t \in \mathbb{R}_0^+$, any $x \in \mathbb{R}^n$, any control input $v : \mathbb{R}_0^+ \rightarrow \mathcal{D}_1 \subseteq \mathbb{R}^m$, and for all possible disturbances $\omega : \mathbb{R}_0^+ \rightarrow \mathcal{D}_2 \subseteq \mathbb{R}^q$, where \mathcal{D}_1 , and \mathcal{D}_2 are compact sets, the following condition is satisfied:*

$$(2.7) \quad \|\xi_{xv}(t)\|_{\mathcal{A}} \leq \beta(\|x\|_{\mathcal{A}}, t),$$

where the point-to-set distance $\|x\|_{\mathcal{A}}$ is defined by $\|x\|_{\mathcal{A}} = \inf_{y \in \mathcal{A}} \|x - y\|$.

When the set \mathcal{A} is a singleton $\{x_0\}$, we speak of an asymptotically stable equilibrium point x_0 rather than a UGAS set. The notion of UGAS for discrete-time control systems is obtained from Definition 2.1 by replacing $t \in \mathbb{R}_0^+$ with $r \in \mathbb{N}_0$.

We recall the following result describing how stability properties are affected by additive disturbances.

Proposition 2.2 ([AMST10]). *Consider the discrete-time linear system:*

$$x[r+1] = Ax[r],$$

and assume that the origin is an asymptotically stable equilibrium point. Then, for any signal $d : \mathbb{N}_0 \rightarrow \mathbb{R}^m$ satisfying $\|d[r]\| \leq b(d)$ for any $r \in \mathbb{N}_0$ and some constant $b(d) \in \mathbb{R}_0^+$, the system:

$$(2.8) \quad x[r+1] = Ax[r] + Bd[r],$$

is UGAS with respect to the set:

$$\mathcal{A} = \{x \in \mathbb{R}^n \mid \|x\| \leq \gamma b(d)\},$$

where γ is given by:

$$(2.9) \quad \gamma = \max_{\theta \in [0, 2\pi[} \left\| (e^{i\theta} I_n - A)^{-1} B \right\|,$$

with $i = \sqrt{-1}$. Moreover, the output $y = Cx$ is guaranteed to converge to the set:

$$(2.10) \quad \mathcal{A}_y = \{y \in \mathbb{R}^p \mid \|y\| \leq \gamma_y b(d)\},$$

with:

$$\gamma_y = \max_{\theta \in [0, 2\pi[} \left\| C (e^{i\theta} I_n - A)^{-1} B \right\|.$$

In control theory, γ_y is known as the \mathcal{L}_2 gain of the control system in (2.8) with the output $y = Cx$. The following proposition follows from Proposition 2.2 and describes the stability properties of linear control systems in (2.6) with respect to disturbance, measurement noise, and implementation errors in the feedback gain and observer dynamic.

Proposition 2.3. *Consider the discrete-time linear system in (2.6). Then for any input e_1 and e_2 satisfying $\|e_1[r]\| \leq b(e_1)$ and $\|e_2[r]\| \leq b(e_2)$ for any $r \in \mathbb{N}_0$ and some constants $b(e_1), b(e_2) \in \mathbb{R}_0^+$, the system is UGAS with respect to the set:*

$$\mathcal{A} = \{x \in \mathbb{R}^n \mid \|x\| \leq \gamma_1 b(e_1) + \gamma_2 b(e_2)\},$$

where γ_1 and γ_2 are given by:

$$\gamma_j = \max_{\theta \in [0, 2\pi[} \left\| (e^{i\theta} I_{2n} - G)^{-1} H_j \right\|, \text{ for } j = 1, 2,$$

with $i = \sqrt{-1}$. Moreover, the output $\mathbb{R}^p \ni y = [C \ 0_{p \times n}] w$ is guaranteed to converge to the set:

$$(2.11) \quad \mathcal{A}_y = \{y \in \mathbb{R}^p \mid \|y\| \leq \gamma_{1y} b(e_1) + \gamma_{2y} b(e_2)\},$$

where γ_{1y} and γ_{2y} are given by:

$$(2.12) \quad \gamma_{jy} = \max_{\theta \in [0, 2\pi[} \left\| [C \ 0_{p \times n}] (e^{i\theta} I_{2n} - G)^{-1} H_j \right\|, \text{ for } j = 1, 2.$$

The error vector e_1 includes disturbance and measurement noise, depending for example on the environment and the quality of the sensors collecting measurements. Hence, the controller designer does not have any control on the value of $b(e_1)$. However, one can reduce the amount of γ_{1y} by appropriately choosing gains K and L . On the other hand, one can reduce the amount of not only γ_{2y} but also $b(e_2)$ by appropriately choosing gains K and L . We use Proposition 2.3 in the following way. Given a feedback gain K and an observer gain L , we compute \mathcal{L}_2 gains γ_{1y} and γ_{2y} and an upper bound $b(e_2)$ on the implementation error e_2 . Then the output of the controlled system (with implementation error) must converge to set \mathcal{A}_y in (2.11). We show later that appropriate choices of gains K and L can shrink the size of the set \mathcal{A}_y and hence, provide a tighter bound on the set to which the output of the system converges.

2.3. LQR-LQG performance. In addition to asymptotic stability, controller designers also consider the *performance* of the controller, that is, of the controllers ensuring asymptotic stability of the origin, one desires the controller that minimizes a given cost function. A common approach for optimal output feedback controller are the *linear quadratic regulator* (LQR) and *linear quadratic Gaussian* (LQG). The LQR cost function to be minimized is given by:

$$(2.13) \quad J_{LQR} = \sum_{r=0}^{+\infty} \{x[r]^T Q x[r] + u[r]^T R u[r]\},$$

for some chosen weight matrices Q and R that are positive definite and of appropriate dimensions.

The LQG cost function to be minimized is given by:

$$(2.14) \quad J_{LQG} = \lim_{r \rightarrow +\infty} \mathbb{E} [\|e[r]\|^2],$$

where \mathbb{E} stands for expected value and e is the estimation error for the control system in (2.4) whose dynamic is given by:

$$(2.15) \quad e[r+1] = x[r+1] - \hat{x}[r+1] = (A_\tau - LC)e[r] + \bar{B}_\tau d[r] - Lv[r].$$

As mentioned before, d and v are assumed to be zero-mean Gaussian noise process (uncorrelated from each other) with covariance matrices:

$$(2.16) \quad \mathbb{E}(d[r]d[r]^T) = \hat{Q}, \quad \mathbb{E}(v[r]v[r]^T) = \hat{R}, \quad \forall r \in \mathbb{N}_0,$$

where \hat{Q} and \hat{R} are some positive semi-definite matrices of appropriate dimensions.

A standard control-theoretic construction rewrites the cost function (2.13) as $J_{LQR} = x[0]^T S(K)x[0]$, where $u = -Kx$, and $S(K) \in \mathbb{R}^{n \times n}$ is a positive definite matrix that is the unique solution for S to the Lyapunov equation:

$$(2.17) \quad (A_\tau - B_\tau K)^T S (A_\tau - B_\tau K) - S + Q + K^T R K = 0,$$

where K is a controller making $A_\tau - B_\tau K$ Hurwitz.¹ See [Hes09] for detailed information. Additionally, we have

$$(2.18) \quad \lambda_{\min}(S(K))\|x[0]\|^2 \leq J_{LQR} \leq \lambda_{\max}(S(K))\|x[0]\|^2,$$

where $\lambda_{\min}(S(K)) \in \mathbb{R}^+$ and $\lambda_{\max}(S(K)) \in \mathbb{R}^+$ are minimum and maximum eigenvalues of $S(K)$, respectively. Therefore, J_{LQR} can be minimized for all possible choice of initial conditions by just minimizing the maximum eigenvalue of $S(K)$. Note that since S is a positive definite matrix, its maximum eigenvalue is equal to its induced 2 norm² $\|S\|$.

Similarly, the cost function (2.14) can be rewritten as $J_{LQG} = \|P(L)\|$, where $P(L) \in \mathbb{R}^{n \times n}$ is a positive definite matrix that is the unique solution for P to the Lyapunov equation:

$$(2.19) \quad (A_\tau - LC) P (A_\tau - LC)^T - P + \bar{B}_\tau \hat{Q} \bar{B}_\tau^T + L \hat{R} L^T = 0,$$

where L is an observer gain making $A_\tau - LC$ Hurwitz. See [Hes09] for more detailed information. Therefore, J_{LQG} can be minimized by just minimizing $\|P(L)\|$.

Note that the optimal feedback $u = -Kx$ minimizing the LQR cost in (2.13) is computed using the deterministic dynamic:

$$x[r+1] = A_\tau x[r] + B_\tau u[r].$$

On the other hand, the optimal gain L minimizing the LQG cost in (2.14) is computed using the stochastic dynamic in (2.15). Thanks to the separation principle for linear control systems [Hes09], one concludes that the overall closed loop system in (2.4) is *UGAS* even though the gains K and L are designed separately.

¹ We call the matrix $A_\tau - B_\tau K$ Hurwitz if its eigenvalues are inside the unit circle, centered at the origin.

² We recall that induced 2 norm of a matrix $A \in \mathbb{R}^{n \times m}$ is given by: $\|A\| = \sqrt{\lambda_{\max}(A^T A)}$.

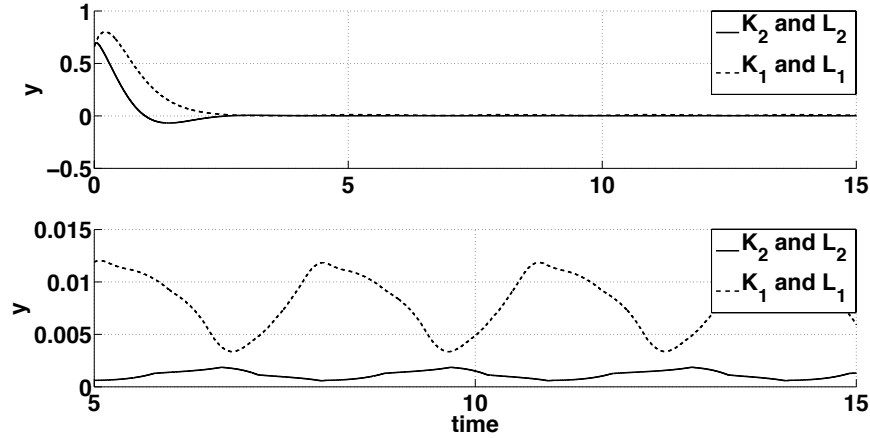


FIGURE 1. Evolution of the output y with initial state $(0.2, 0.2)^T$ for the pair of gains K_1 , L_1 and K_2 , L_2 using 16-bit implementation.

2.4. The effect of errors. Example We now present a simple motivating example showing how different choice of controllers result in different steady state errors due to their fixed-point implementations, yet providing approximately the same LQR-LQG performance. Consider a simple physical model of a bicycle, borrowed from [AM08]. The dynamics of the system is given by:

$$(2.20) \quad \begin{cases} \begin{bmatrix} \dot{\xi}_1 \\ \dot{\xi}_2 \end{bmatrix} = \begin{bmatrix} 0 & \frac{g}{h} \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \xi_1 \\ \xi_2 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} (v + \omega), \\ \eta = \begin{bmatrix} \frac{av_0}{bh} & \frac{v_0^2}{bh} \end{bmatrix} \begin{bmatrix} \xi_1 \\ \xi_2 \end{bmatrix} + \nu, \end{cases}$$

where ξ_1 is the steering angular velocity, ξ_2 is the steering angle, η is the roll angle, v is the torque applied to the handle bars, $g = 9.8m/s^2$ is the acceleration due to gravity, $h = 1.5m$ is the height of the center of mass, $v_0 = 2m/s$ is the velocity of the bicycle at the rear wheel, $a = 0.5m$ is the distance of the center of mass from a vertical line through the contact point of the rear wheel and $b = 1m$ is the wheel base.

The control objective is to design a feedback gain $K \in \mathbb{R}^{1 \times 2}$ and an observer gain $L \in \mathbb{R}^{2 \times 1}$ such that the feedback control law $u = -K\hat{x}$, where $\hat{x} = [\hat{x}_1, \hat{x}_2]^T$ is the state of the observer in (2.3), makes the closed loop system UGAS. By choosing the matrices $Q = I_2$ and $R = 1$ inside the LQR cost function and $\hat{Q} = 1$ and $\hat{R} = 1$ in (2.16), the feedback and observer gains minimizing the LQR and LQG costs are given by $K_1 = [5.1538, 12.9724]$, and $L_1 = [0.0317, 0.0118]^T$, respectively. Consider a second pair of feedback and observer gains given by $K_2 = [3.0253, 12.6089]$ and $L_2 = [0.0132, 0.1021]^T$. For the initial condition $x = (0.2, 0.2)^T$, the value of the LQR cost function is 264.1908 for feedback gain K_1 and 284.1578 for K_2 . Moreover, the value of the LQG cost function is 0.0229 for observer gain L_1 and 0.0246 for L_2 . That is, the gains K_2 and L_2 give cost functions about 7% greater than the optimal gains K_1 and L_1 .

We now show how different choice of feedback and observer gains result in different fixed-point implementation errors. For now, let us assume that $\omega(t) = 0$ and $\nu(t) = 0$, for any $t \in \mathbb{R}_0^+$. In Figure 1, we show the output of the closed-loop system starting from the initial condition $x = (0.2, 0.2)^T$, when the feedback gain and observer dynamic are implemented using 16-bit fixed-point representation. As can be observed from Figure 1, the output of the controlled system does not converge to the equilibrium point at the origin because of the fixed-point implementation error in the controllers. Furthermore, the practical stability region using gains K_2 and L_2 is much smaller than the one using gains K_1 and L_1 .

Using bounds on implementation errors for the two controllers (described in Section 3) and Proposition 2.3, we can prove that the output of the system with feedback and observer gains K_1 and L_1 (resp. K_2 and L_2)

converges to a ball centered at the origin with radius 0.5486 (resp. 0.0513), whenever the output of the system and the state of the observer take values in the interval $[-1, 1]$ and the feedback gain and observer dynamic are implemented using 16-bit fixed-point implementation. As can be seen, given a 16-bit implementation, feedback and observer gains K_2 and L_2 may be preferred to gains K_1 and L_1 because they have guaranteed bounds on practical stability region that is 10 times smaller than gains K_1 and L_1 and provide approximately similar performance. If one considers the effect of disturbance and measurement noise, it can be proved that the output of the system with feedback and observer gains K_1 and L_1 (resp. K_2 and L_2) converges to a ball centered at the origin with radius $5.0489b(e_1) + 0.5486$ (resp. $2.5341b(e_1) + 0.0513$), where $b(e_1)$ is an upper bound on the size of the vector e_1 introduced in (2.6).

Optimization objectives The above example suggests that the control design should optimize for the following objectives: the LQR and the LQG costs for performance, error caused by disturbance and measurement noise, and the implementation error given by a fixed-precision encoding. Accordingly, we define a cost function that is weighted sum of the four factors:

$$(2.21) \quad \mathcal{J}(K, L) = w_1 \frac{\|S(K)\|}{\|S^*\|} + w_2 \frac{\|P(L)\|}{\|P^*\|} + w_3 \frac{\gamma_{1y}}{\gamma_{1y}^*} + w_4 \frac{\gamma_{2y}b(e_2)}{\gamma_{2y}^*b^*(e_2)},$$

where w_1, \dots, w_4 are weighting factors, S^* and P^* are matrices, computed from Lyapunov equations in (2.17) and (2.19) using standard LQR and LQG gains (K_{LQR} and L_{LQG}), γ_{1y} and γ_{2y} (resp. γ_{1y}^* and γ_{2y}^*) are the \mathcal{L}_2 gains in (2.12) using feedback and observer gains K and L (resp. K_{LQR} and L_{LQG}) and $b(e_2)$ (resp. $b^*(e_2)$) is the bound on the implementation error of given feedback and observer gains K and L (resp. K_{LQR} and L_{LQG}). Minimizing the terms γ_{1y} and $\gamma_{2y}b(e_2)$ inside (2.21) results in a tighter bound on the set \mathcal{A}_y in Proposition 2.3. Since the four factors in (2.21) have different scales, we normalized them by their values using the standard gains K_{LQR} and L_{LQG} . The designer can choose w_1, \dots, w_4 based on the priorities on LQR and LQG performances and steady state error. Our objective is to find feedback and observer gains that minimize the cost function \mathcal{J} .

We focus on implementation errors arising out of fixed-precision arithmetic. The bound $b(e_2)$ is computed using the strategy, explained in Section 3. Note that the cost function \mathcal{J} is not necessarily convex with respect to the feedback and observer gains K and L . Therefore, the proposed design strategy cannot be formulated as a convex optimization problem. We use a heuristic stochastic optimization approach to find feedback and observer gains K and L minimizing \mathcal{J} .

In our exposition, we consider the plant model to be precise, and only consider quantization effects as the source of error. Our methodology can consider both additive and multiplicative uncertainties in the plant model as well [GL94]. We can take those uncertainties into account by adding appropriate extra terms to the cost function in (2.21) using the results provided in [ZSKG09, ZKGS07]. We omit the details for simplicity.

3. COMPUTING QUANTIZATION ERROR

In this section we show how to compute a bound on the fixed-point implementation error for given feedback and observer gains K and L . We assume that the outputs of the controlled system and the state of the observer are restricted to compact subsets $Y \subset \mathbb{R}^p$ and $\hat{X} \subset \mathbb{R}^n$, respectively.

3.1. Best fixed-point implementation. A *fixed-point representation* of a real number is a triple $\langle s, n, m \rangle$ consisting of a *sign bit* $s \in \{\mathbf{s}, \mathbf{u}\}$ (for *signed* and *unsigned*), a *length* $n \in \mathbb{N}$, and a *length of the fractional part* $m \in \mathbb{N}$. The length of the integer part is $n - m - 1$. Intuitively, a real number is represented using n bits, of which m bits are used to store the fractional part. Clearly, the largest integer portion has to fit in $n - m - 1$ bits.

A variable with a fixed-point type is represented as an integer. We associate an integer variable \hat{x} with the fixed-point representation of a real variable x . An integer variable \hat{x} that represents a fixed-point variable with type $\langle \mathbf{u}, n, m \rangle$ can be interpreted as the rational number $2^{-m}\hat{x}$. We deal with a signed number by separately

tracking the sign and the magnitude, performing the operations on the magnitudes using unsigned arithmetic, and finally putting the appropriate sign bits back.

An operation using real arithmetic may have different fixed-point implementation depending on how many bits are allocated to hold the integer part and the fraction part of the variables. Allocating fewer number of bits than required to hold the integer part may lead to overflow. On the other hand, if more than the required number of bits are allocated to the integer part, the quantization error increases due to assigning few bits to the fractional part. When we compare fixed-point implementation of different controllers, we first synthesize the best possible implementation of a controller.

Let us fix the number of bits to be n for the implementation of a controller. With n bits, let b be the upper bound on the quantization error in a fixed-point implementation I of a controller for a given range for the inputs. The fixed-point implementation I is the *best implementation* if there does not exist another implementation I' using n bits, for which the upper bound on the quantization error is b' and $b' < b$.

If the ranges of the variables in the real arithmetic computation can be computed exactly, it is possible to synthesize the best fixed-point implementation. In the best fixed-point implementation, the number of bits allocated to the integer part is just enough to hold the integer part of any value in that range. For example, if the range of a variable is $[-35.55, 48.72]$, the datatype for the variable in the best 16-bit fixed-point representation is $\langle 1, 16, 9 \rangle$.

The range computation problem of variable y in an operation $y = f(x_1, \dots, x_n)$ involves solving a maximization and a minimization problem where f is the objective function and the ranges on x_1, \dots, x_n form the set of constraints. If the function f is convex, the range of y can be computed exactly, and also it is straight-forward to find the best fixed-point implementation for the operation.

3.2. Error bound computation. We apply mixed-integer linear programming based optimization technique to find out the error bound between a computation in real arithmetic and its best fixed-point implementation. Suppose we have an arithmetic operation $s : a = b \text{ op } c$, where $\text{op} \in \{+, -, *\}$. If $\text{op} = *$, then either b or c is a constant. If $\text{op} = +$ or $\text{op} = -$, then b and c can both be variables. We associate an integer variable \hat{x} with the fixed-point representation of a real variable x . Let the range of the values for a and b and c are $[l_a, u_a]$, $[l_b, u_b]$, and $[l_c, u_c]$, respectively. Let the fixed-point representation of a , b and c are $\langle \mathbf{s}, n_a, m_a \rangle$, $\langle \mathbf{s}, n_b, m_b \rangle$, and $\langle \mathbf{s}, n_c, m_c \rangle$, respectively. Let $b(e_b)$ and $b(e_c)$ be bounds on the quantization errors of b and c , respectively. The optimization problem to find the bound on the error is given by:

$$\begin{aligned}
 & \text{maximize} && |a - 2^{-m_a} \hat{a}| \\
 & \text{Subject to} && l_a \leq a \leq u_a \\
 & && l_b \leq b \leq u_b \\
 (3.1) \quad & && |b - 2^{-m_b} * \hat{b}| \leq b(e_b) \\
 & && |c - 2^{-m_c} * \hat{c}| \leq b(e_c) \\
 & && a = b \text{ op } c \\
 & && \Phi(\text{fp}(s))
 \end{aligned}$$

where $\text{fp}(s)$ is the fixed-point representation of the statement s and $\Phi(\mathbf{s})$ denotes a logical formula that relates the inputs and outputs of the fixed-point representation \mathbf{s} . Technically, Φ is the *strongest postcondition* [Win93] of \mathbf{s} with respect to *true*. We compute Φ using an arithmetic encoding of a fixed-point computation [AMST10]. Here we illustrate the computation of the strongest postcondition Φ using an example.

Example. Suppose we have the following arithmetic operation

$$s : y = -7.2479 * x .$$

Assume the compact set for x is $[-1, 1]$. The fixed-point expression corresponding to s in the best fixed-point implementation is

$$\text{fp}(s) : -\hat{y} = (-115 * \hat{x}) \gg 6 .$$

The strongest postcondition $\Phi(\text{fp}(s))$ of $\text{fp}(s)$ is given by:

$$\begin{aligned}\Phi(\text{fp}(s)) := & \quad tmp = -115 * \hat{x} \wedge \\ & tmp \geq 0 \rightarrow tmp1 = tmp \wedge \\ & tmp < 0 \rightarrow tmp1 = -tmp \wedge \\ & tmp1 = 2^6 * divisor + remainder \wedge \\ & remainder \geq 0 \wedge remainder < 2^6 \wedge \\ & tmp \geq 0 \rightarrow \hat{y} = divisor \wedge \\ & tmp < 0 \rightarrow \hat{y} = -divisor ,\end{aligned}$$

where tmp , $tmp1$, $divisor$, and $remainder$ are integer variables.

Depending on the arithmetic operation, we need to solve at most 4 instances of mixed integer linear programming problem to solve the optimization problem in (3.1), and the maximum among all of them gives the bound on the error in the fixed-point implementation.

We use the above technique to compute the bound on the error in one operation in the fixed-point implementation of a gain. The implementation of a gain involves a series of arithmetic operations. We compute the error bound for the output of one arithmetic operation at a time. Let $s : a = b \text{ op } c$ is an arithmetic operation in the implementation of a gain. In the arithmetic operation, b and c may either be a constant, a state variable or a temporary variable which captures the result of some previous operation. If b (or c) represents a constant, and the fixed-point representation contains m bits for the fraction part, then the error in the fixed point representation is bounded by $\frac{1}{2^m}$. If b (or c) represents a state variable, then the fixed-point datatype can be determined from the given compact set for the state, and the fixed-point datatype can be determined accordingly. Then the error in the fixed-point representation is bounded by $\frac{1}{2^m}$, where m is the number of bits to represent the fraction part in the fixed-point datatype of the variable. If b (or c) is a temporary variable used to hold the result of an earlier computation, then the range and error bound for the variable is already known.

4. OPTIMAL CONTROLLER SYNTHESIS

We now describe our controller synthesis algorithm that minimizes the cost function (2.21) combining LQR and LQG performance, disturbance, measurement noise and implementation errors. Since the cost function is non-convex, we use a stochastic local search technique.

4.1. Particle swarm optimization (PSO). PSO is a stochastic local search approach. It maintains a set of potential solutions (called “particles”) in a compact d dimensional search space $D = \prod_{j=1}^d [y_{\min}^j, y_{\max}^j] \subset \mathbb{R}^d$, minimizing a given cost function. The particles move in this space according to their velocity. Each particle, indexed by $i \in \mathbb{N}$, has a position $y_i \in \mathbb{R}^d$, changing between y_{\min} and y_{\max} , and a velocity vector $v_i \in \mathbb{R}^d$, changing between some vectors v_{\min} and v_{\max} . The terms v_{\min} and v_{\max} are often set to the maximum dynamic range of the variables on each dimension [ZKGSP09]: $-v_{\min}^j = v_{\max}^j = |y_{\max}^j - y_{\min}^j|$. Every particle remembers its own best position (i.e., the lowest value of the cost function achieved so far by this particle) in a vector $P_i \in \mathbb{R}^d$. The best position with respect to the cost function among all of the particles so far is stored in a vector $P_g \in \mathbb{R}^d$.

PSO updates the positions and velocities of all particles iteratively. The new velocity and position for particle i are determined as:

$$(4.1) \quad v_i^{l+1} = w^l v_i^l + c_1 r_1 (P_i^l - y_i^l) + c_2 r_2 (P_g^l - y_i^l) ,$$

$$(4.2) \quad y_i^{l+1} = y_i^l + v_i^{l+1} ,$$

where the superscript l denotes the iteration number, the subscript $i = 1, \dots, N$ denotes the index of the particle, and N is the number of particles. The constant w^l in (4.1) is updated using the inertia weight

approach [EKG12] as the following:

$$(4.3) \quad w^l = \max \left\{ w_{\min}, w_{\max} - \frac{w_{\max} - w_{\min}}{l_{\max}}(l - 1) \right\},$$

where w_{\max} and w_{\min} are adjusted to 1 and $\frac{c_1 + c_2}{2} - 1$ and l_{\max} is the maximum number of iterations. The constants c_1 and c_2 in (4.1) are the acceleration constants, influencing the convergence speed of particles toward its own and global best positions and set to 0.5 and 1, respectively [EKG12]. The constants r_1 and r_2 in (4.1) are uniformly distributed random numbers on the interval $[0, 1]$.

4.2. Overall algorithm. The PSO algorithm is used to search for feedback and observer gains $K \in \mathbb{R}^{m \times n}$ and $L \in \mathbb{R}^{n \times p}$ for the control system (2.5), minimizing (2.21). Note that a particle in PSO represents a feedback and an observer gain K and L , respectively, moving in an $m \times n + n \times p$ dimensional search space. To discard those gains that make the controlled system unstable, we penalize unstable gains by including a penalty term \tilde{P} in the cost function such that $\tilde{P} = 0$ if $A_\tau - B_\tau K$ and $A_\tau - LC$ are Hurwitz and $\tilde{P} = +\infty$ otherwise. The cost function for PSO is then $F(K, L) = \mathcal{J}(K, L) + \tilde{P}(K, L)$.

The design steps can be summarized as the following:

- (1) Initialize positions of N feedback gains K_i and observer gains L_i by K_{LQR} and L_{LQG} , respectively, and uniformly randomly initialize their velocities, $i = 1, \dots, N$, where N is the number of particles.
- (2) Given any initial feedback gain K_i and observer gain L_i , compute the cost function $F(K_i, L_i)$. To compute \tilde{P} , check if $A_\tau - B_\tau K$ and $A_\tau - LC$ are Hurwitz. There are some steps to compute \mathcal{J} . First compute $S(K_i)$ and $P(L_i)$ by solving the Lyapunov equations (2.17) and (2.19), respectively, and find their induced 2 norm. Second, compute the \mathcal{L}_2 gains γ_{1y} and γ_{2y} . Third, compute $b(e_2)$ by solving the optimization problems from Section 3.
- (3) Compare $F(K_i, L_i)$ to its own best position P_i so far and the global best position P_g so far. If $F(K_i, L_i)$ is less than the previous personal best (resp. the global best), update the best position (resp. the global best) to K_i and L_i .
- (4) Modify the velocity and position of each pair K_i and L_i according to (4.1) and (4.2).
- (5) If the number of iterations, denoted by l , reaches the maximum, denoted by l_{\max} , or the value of F does not change for the global best position P_g for 50 consecutive iterations up to error 10^{-6} then go to Step (6), otherwise go to Step (2);
- (6) The latest P_g is the optimal controller.

5. EXTENSION: PID CONTROLLERS

PID controllers are a common class of controllers in many industries, such as automotive, power systems, servomotors, and so on. We now extend the analysis of Section 2 to PID controllers. A PID controller generalizes a proportional feedback controller, and includes three terms: proportional, integrator, and differentiator. For an input v , the output η of the PID controller is computed as follows:

$$(5.1) \quad \eta(t) = K_P v(t) + K_I \int_0^t v(s) ds + K_D \frac{dv(t)}{dt}, \quad \forall t \in \mathbb{R}_0^+,$$

where K_P , K_I , and K_D are called proportional, integrator, and differentiator gains, respectively. To describe the mismatch between the PID specifications and its software implementation, we consider the discrete-time version of (5.1). A common way of discretizing an integrator is based on the trapezoidal approximation. An integrator term:

$$\eta(t) = \int_0^t v(s) ds, \quad \forall t \in \mathbb{R}_0^+,$$

can be discretized as follows:

$$(5.2) \quad y[r + 1] = y[r] + \frac{\tau}{2} (u[r + 1] + u[r]), \quad \forall r \in \mathbb{N}_0,$$

where τ is the sampling time, $y[r] = \eta(r\tau) + e_1$ and $u[r] = v(r\tau)$, for any $r \in \mathbb{N}_0$. A common way of discretizing a differentiator, is based on the backward Euler method. A differentiator term:

$$\eta(t) = \frac{dv(t)}{dt}, \quad \forall t \in \mathbb{R}_0^+,$$

can be discretized as follows:

$$(5.3) \quad y[r+1] = \frac{u[r+1] - u[r]}{\tau}, \quad \forall r \in \mathbb{N}_0,$$

where $y[r] = \eta(r\tau) + e_2$ and $u[r] = v(r\tau)$, for any $r \in \mathbb{N}_0$. By using fast sampling time assumption, we can ignore the errors e_1 and e_2 in the discretized versions of integrator and differentiator in comparison with quantization errors. To follow the same analysis as in Section 2, we need a state space realization of PID controller. By resorting to results in control classic [Kai80] and using the discretization rules in (5.2) and (5.3), the state space realization of discretized PID controller with input $\hat{u}[r]$ and output $\hat{y}[r]$ is obtained as follows:

$$(5.4) \quad \begin{cases} \hat{x}[r+1] = \hat{A}\hat{x}[r] + \hat{B}\hat{u}[r], \\ \hat{y}[r] = \hat{C}\hat{x}[r] + \hat{D}\hat{u}[r], \end{cases}$$

where

$$\hat{A} = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}, \quad \hat{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \hat{C} = \begin{bmatrix} K_D & K_I\tau - \frac{K_D}{\tau} \end{bmatrix}, \quad \hat{D} = \left(K_P + \frac{K_I\tau}{2} + \frac{K_D}{\tau} \right).$$

Without loss of generality, consider a single-input ($m = 1$) single-output ($p = 1$) discrete-time linear control system of the form:

$$\begin{cases} x[r+1] = Ax[r] + Bu[r], \\ y[r] = Cx[r]. \end{cases}$$

Since the input of the PID controller is equal to the negative of the output of the plant ($\hat{u} = -y$) because of negative feedback and the output of the PID controller is equal to the input of the plant ($u = \hat{y}$), one obtains:

$$(5.5) \quad \begin{cases} x[r+1] = (A - B\hat{D}\hat{C})x[r] + B\hat{C}\hat{x}[r], \\ \hat{x}[r+1] = -\hat{B}\hat{C}x[r] + \hat{A}\hat{x}[r]. \end{cases}$$

Similar to what explained in Section 2, by fixed-point implementation of the PID controller, one gets the following overall dynamic:

$$(5.6) \quad \begin{cases} x[r+1] = (A - B\hat{D}\hat{C})x[r] + B\hat{C}\hat{x}[r] + Be_{q2}, \\ \hat{x}[r+1] = -\hat{B}\hat{C}x[r] + \hat{A}\hat{x}[r] + e_{q1}, \end{cases}$$

where e_{q1} and e_{q2} are quantization errors in computing the PID controller. Now, we can use the same strategy, as explained in Subsection 4.2, to design parameters K_P , K_I , and K_D of PID controllers minimizing a performance-based cost function as well as the effect of quantization error. For example, one can consider:

$$(5.7) \quad \mathcal{J}(K_P, K_I, K_D) = \frac{w_1}{\text{PM}} + \frac{w_2}{\text{GM}} + w_3\gamma(b(e_{q1}) + b(e_{q2})),$$

where PM and GM are phase and gain margins, w_1, w_2, w_3 are weighting factors, γ is the \mathcal{L}_2 gain of the linear control system (5.6) and $b(e_{q1})$ and $b(e_{q2})$ are the bounds on the implementation errors e_{q1} and e_{q2} . Note that control over PM and GM guarantees robust stability of the closed-loop systems [Hes09]. The phase and gain margins measure the system's tolerance to the time delay and the steady state gain, respectively.

Control systems	# bits	Synthesized gains		Time cost
		K	L	
Bicycle	16	[3.0253 12.6089]	[0.0132 0.1021] ^T	1h36m41s
DC motor position	16	[0.1129 0.0211 0.0093]	[0.0390 0.3700 -0.0175] ^T	1h39m06s
Pitch angle control	32	[-0.1202 42.5655 1.0001]	[0.0001 0 0.0017] ^T	8h31m53s
Inverted pendulum	32	[-1.5362 -2.0254 16.5192 2.7358]	$\begin{bmatrix} 0.0017 & 0.0021 & 0.0012 & 0 \\ 0.0001 & 0.0018 & 0.0122 & 0.0770 \end{bmatrix}^T$	9h54m17s
Batch reactor process	16	$\begin{bmatrix} 0.0583 & 0.9093 & 0.3258 & 0.8721 \\ -2.4638 & -0.0504 & -1.7099 & 1.1653 \end{bmatrix}$	$\begin{bmatrix} 0.0774 & -0.0022 & 0.0267 & 0.0356 \\ -0.0103 & 0.0227 & 0.0398 & 0.0001 \end{bmatrix}^T$	3h08m29s

TABLE 1. Synthesized gains and required time for synthesizing them.

Control systems	lub of LQR cost		LQG cost		Steady state error	
	LQR	Synthesized K	LQG	Synthesized L	LQR-LQG	Synthesized gains
Bicycle	$3956.3\ x\ ^2$	$4331.7\ x\ ^2$	0.0229	0.0246	$5.0489b(e_1)+0.5486$	$2.5341b(e_1)+0.0513$
DC motor position	$1001.6\ x\ ^2$	$1376.7\ x\ ^2$	36.6315	36.6731	$30.5666b(e_1)+0.16$	$15.4216b(e_1)+0.011$
Pitch angle control	$2.9732 \times 10^6\ x\ ^2$	$2.9887 \times 10^6\ x\ ^2$	0.0013	0.0018	$2.6781b(e_1)+0.4746$	$1.4453b(e_1)+0.0807$
Inverted pendulum	$4.2988 \times 10^4\ x\ ^2$	$5.3471 \times 10^4\ x\ ^2$	0.3600	0.3897	$83.4217b(e_1)+0.0432$	$30.3801b(e_1)+0.0086$
Batch reactor process	$223.1773\ x\ ^2$	$223.1825\ x\ ^2$	0.0731	0.0949	$2.9309b(e_1)+0.4194$	$2.1216b(e_1)+0.1642$

TABLE 2. Least upper bound (lub) on the LQR cost (2.13), for a given initial condition x , the LQG cost (2.14), and the Euclidean norm of the steady state error for the LQR-LQG and the synthesized gains.

6. EXPERIMENTAL RESULTS

We implemented the algorithm presented in Section 4.2 in Matlab. We use a PSO function in Matlab, introduced in [EKG12]. We implemented a static analyzer in Ocaml that synthesizes the best fixed-point program and computes the bound on the fixed-point implementation error for given feedback and observer gains K and L , respectively. The tool gets the number of the bits in the fixed-point datatype, compact subsets $Y \subset \mathbb{R}^p$ and $\hat{X} \subset \mathbb{R}^n$, and feedback and observer gains K and L , respectively, as inputs. The optimization problems in computing the error bound are solved using the mixed-integer linear programming tool `lp_solve` [lps].

We applied the proposed controller synthesis approach to a number of linear control systems. All the experiments were done on a laptop with CPU Intel Core 2 Duo at 2.4 GHz. In all of the experiments, the number of the particles in PSO is $N = 24$, the maximum number of iterations is set to $l_{\max} = 100$, and we choose the matrices $Q = I_n$, and $R = I_m$ in (2.13) and $\hat{Q} = I_q$, and $\hat{R} = I_p$ in (2.16). The value of l_{\max} was chosen in such a way that appropriate gains are obtained in terms of the cost function (2.21) (or (5.7)) for all control systems. Moreover, we assume that the search space is $D = \prod_{i=1}^{n \times m + n \times p} [-150, 150] \subset \mathbb{R}^{n \times m + n \times p}$ that is large enough and contains the standard LQR and LQG gains for all the examples. Furthermore, without loss of generality, we work on the compact subsets $Y = \prod_{i=1}^p [-1, 1] \subset \mathbb{R}^p$ and $\hat{X} = \prod_{i=1}^n [-1, 1] \subset \mathbb{R}^n$. All constants and variables are expressed in SI units.

Our unstable examples include a bicycle [AM08], a DC motor position control [cmu], a pitch angle control [cmu], an inverted pendulum [cmu], a batch reactor process [GL94] and another inverted pendulum for PID synthesis [cmu]. See Table 1 and 2 for experimental results. Note that for those examples for which 32-bit implementation is chosen, the 16-bit one provides a stability region which is even larger than the range of the variables inside the controller. As can be seen from Table 2, in comparison with the conventional LQR-LQG approach, the proposed synthesis approach in this paper worsens the LQR and LQG performances by at most 1.37 times (for DC motor position) and 1.38 times (for Pitch angle control), respectively. However, the proposed synthesis approach improves the size of the region of practical stability due to quantization error by at least 2.55 times. For certain examples, the improvement goes beyond the factor of 10. For bicycle and DC

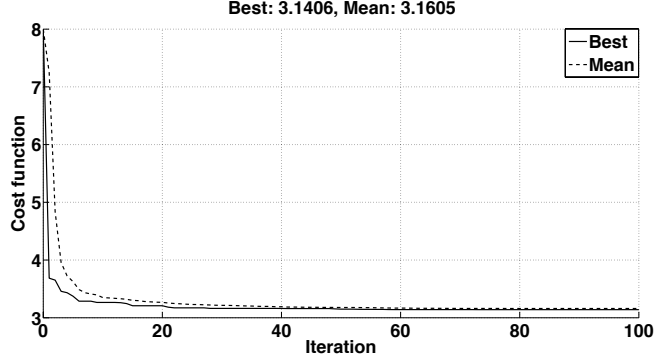


FIGURE 2. Cost of the best particle and average cost of all population vs iteration.

motor position, the region of practical stability due to quantization error improves by a factor of 10.69 and 14.55, respectively.

The detailed description of the systems are available as follows.

Bicycle The model of a bicycle is shown in (2.20). The weighting factors in (2.21) are chosen as $w_1 = w_2 = w_3 = 1$ and $w_4 = 5$. The results of the LQR, LQG and the proposed method are shown in Tables 1 and 2. Figure 2 shows how the value of the cost function improves with the number of iteration. The figure shows how the value of the cost function monotonically decreases with the number of iterations. The fixed-point C code for the synthesized controller is shown in Figure 3.

DC motor position control The dynamic of a DC motor position control, borrowed from [cmu], is given by:

$$\left\{ \begin{array}{l} \begin{bmatrix} \dot{\xi}_1 \\ \dot{\xi}_2 \\ \dot{\xi}_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & \frac{-b}{J} & \frac{K}{J} \\ 0 & \frac{-K}{L} & \frac{-R}{L} \end{bmatrix} \begin{bmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{1}{L} \end{bmatrix} (v + \omega), \\ \eta = [1 \ 0 \ 0] \begin{bmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \end{bmatrix} + \nu, \end{array} \right.$$

where ξ_1 is the angle of the motor's shaft, ξ_2 is the angular velocity of the motor's shaft, ξ_3 is the armature current, $b = 3.508 \times 10^{-6}$ is the damping ratio of the mechanical system, $J = 3.228 \times 10^{-6}$ is the moment of inertia of the rotor, $K = 0.027$ is the electromotive force constant, $R = 4$ is the electric resistance, $L = 2.75 \times 10^{-6}$ is the electric inductance and v is the source voltage. The weighting factors in (2.21) are chosen as $w_1 = w_2 = w_3 = 1$ and $w_4 = 5$. The LQR and LQG gains are given by $K_{LQR} = [0.4055 \ 0.3782 \ 0.0022]$ and $L_{LQG} = [0.0288 \ 0.3858 \ -0.0026]^T$ and the gains, computed by the proposed approach in this paper, are given in Table 1. The detailed results are shown in Tables 1 and 2.

Pitch control The dynamic of the longitudinal motion of an aircraft, borrowed from [cmu], is given by:

$$\left\{ \begin{array}{l} \begin{bmatrix} \dot{\xi}_1 \\ \dot{\xi}_2 \\ \dot{\xi}_3 \end{bmatrix} = \begin{bmatrix} -0.313 & 56.7 & 0 \\ -0.0139 & -0.426 & 0 \\ 0 & 56.7 & 0 \end{bmatrix} \begin{bmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \end{bmatrix} + \begin{bmatrix} 0.232 \\ 0.0203 \\ 0 \end{bmatrix} (v + \omega) \\ \eta = [0 \ 0 \ 1] \begin{bmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \end{bmatrix} + \nu, \end{array} \right.$$

where ξ_1 is the angle of attack, ξ_2 is the pitch rate, ξ_3 is the pitch angle, and v is elevator deflection angle. The weighting factors in (2.21) are chosen as $w_1 = w_2 = w_3 = 1$ and $w_4 = 5$. The LQR and LQG gains are given

```

float output(float yin)
{
    static int x1 = x1_0; // fixdt(1,16,14)
    static int x2 = x2_0; // fixdt(1,16,14)
    int x1_new;           // fixdt(1,16,14)
    int x2_new;           // fixdt(1,16,14)
    int u;                // fixdt(1,16,11)

    // Intermediate variables
    int Gain1;            // fixdt(1,16,15)
    int Gain2;            // fixdt(1,16,15)
    int Gain3;            // fixdt(1,16,15)
    int Add1;             // fixdt(1,16,14)
    int Gain4;            // fixdt(1,16,15)
    int Gain5;            // fixdt(1,16,15)
    int Gain6;            // fixdt(1,16,15)
    int Add2;             // fixdt(1,16,15)
    int Gain7;            // fixdt(1,16,13)
    int Gain8;            // fixdt(1,16,11)

    y = convert_to_fixedpoint(yin);
    Gain1 = (31499 * x1) >> 14;
    Gain2 = (-3145 * x2) >> 14;
    Add1 = (Gain1 + Gain2) >> 1;
    Gain3 = (432 * y) >> 14;
    x1_new = ((Add1 << 1) + Gain3) >> 1;
    Gain4 = (-1907 * x1) >> 14;
    Gain5 = (23835 * x2) >> 14;
    Add2 = Gain4 + Gain5;
    Gain6 = (3345 * y) >> 14;
    x2_new = (Add1 + Gain6) >> 1;
    Gain7 = (24783 * x1_new) >> 14;
    Gain8 = (25823 * x2_new) >> 14;
    u = (Gain7 + (Gain8 << 2)) >> 2;
    return(float(u));
}

```

FIGURE 3. synthesized fixed-point controller C code for Bicycle.

by $K_{LQR} = [-0.1141 \ 49.1428 \ 0.9995]$ and $L_{LQG} = 10^{-3} \times [0.6407 \ 0.0039 \ 0.6655]^T$ and the gains, computed by the proposed approach in this paper, are given in Table 1. The detailed results are shown in Tables 1 and 2.

Inverted pendulum Consider a simple physical model of an inverted pendulum on a cart, borrowed from [cmu]. The dynamics of the system is given by:

$$\left\{ \begin{aligned} \begin{bmatrix} \dot{\xi}_1 \\ \dot{\xi}_2 \\ \dot{\xi}_3 \\ \dot{\xi}_4 \end{bmatrix} &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-(I+ml^2)b}{I(M+m)+Mml^2} & \frac{m^2gl^2}{I(M+m)+Mml^2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{-mlb}{I(M+m)+Mml^2} & \frac{mgl(M+m)}{I(M+m)+Mml^2} & 0 \end{bmatrix} \begin{bmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \\ \xi_4 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{I+ml^2}{I(M+m)+Mml^2} \\ 0 \\ \frac{ml}{I(M+m)+Mml^2} \end{bmatrix} v + \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \omega, \\ \eta &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \\ \xi_4 \end{bmatrix} + \nu, \end{aligned} \right.$$

where ξ_1 , and ξ_2 are the position and velocity of the cart, respectively, ξ_3 , and ξ_4 are the angular position and velocity of the mass to be balanced, v is the applied force to the cart, $g = 9.8$ is the acceleration due to gravity, $l = 0.3$ is the length of the rod, $m = 0.2$ is the mass of the system to be balanced, $M = 0.5$ is the mass of the cart, $b = 0.1$ is the coefficient of friction of the cart, and $I = 0.006$ is the inertia of the pendulum. The weighting factors in (2.21) are chosen as $w_1 = w_2 = w_3 = 1$ and $w_4 = 5$. The LQR and LQG gains are

given by $K_{LQR} = [-0.9929 \ -2.0276 \ 20.2819 \ 3.9126]$ and

$$L_{LQG} = \begin{bmatrix} 0.0016 & 0.0011 & 0.0007 & 0.0034 \\ 0.0007 & 0.0051 & 0.0111 & 0.0618 \end{bmatrix}^T,$$

and the gains, computed by the proposed approach in this paper, are given in Table 1. The detailed results are shown in Tables 1 and 2.

Batch reactor process Consider an unstable batch reactor process, borrowed from [GL94]. The dynamic of the system is given by:

$$\begin{cases} \begin{bmatrix} \dot{\xi}_1 \\ \dot{\xi}_2 \\ \dot{\xi}_3 \\ \dot{\xi}_4 \end{bmatrix} = \begin{bmatrix} 1.38 & -0.2077 & 6.715 & -5.676 \\ -0.5814 & -4.29 & 0 & 0.675 \\ 1.067 & 4.273 & -6.654 & 5.893 \\ 0.048 & 4.273 & 1.343 & -2.104 \end{bmatrix} \begin{bmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \\ \xi_4 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 5.679 & 0 \\ 1.136 & -3.146 \\ 1.136 & 0 \end{bmatrix} v + \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \omega, \\ \eta = \begin{bmatrix} 1 & 0 & 1 & -1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \\ \xi_4 \end{bmatrix} + \nu. \end{cases}$$

The weighting factors in (2.21) are chosen as $w_1 = w_3 = 1$, $w_2 = 2$, and $w_4 = 5$. The LQR and LQG gains are given by:

$$\begin{aligned} K_{LQR} &= \begin{bmatrix} 0.0376 & 0.9157 & 0.3262 & 0.8226 \\ -2.4884 & -0.0734 & -1.7461 & 1.1438 \end{bmatrix}, \\ L_{LQG} &= \begin{bmatrix} 0.0447 & -0.0003 & 0.0170 & 0.0127 \\ 0 & 0.0020 & 0.0058 & 0.0059 \end{bmatrix}^T, \end{aligned}$$

and the gains, computed by the proposed approach in this paper, are given in Table 1. The detailed results are shown in Tables 1 and 2.

PID controller In this example, we provide a PID controller for an inverted pendulum whose dynamic is given by a transfer function. Consider the transfer function of an inverted pendulum, borrowed from [cmu], given by:

$$(6.1) \quad \frac{\Phi(s)}{U(s)} = \frac{\frac{ml}{q}s}{s^3 + \frac{b(I+ml^2)}{q}s^2 - \frac{(M+m)mgl}{q}s - \frac{bmgl}{q}},$$

where $q = (M + m)(I + ml^2) - (ml)^2$, output ϕ is the angular position of the mass to be balanced, input v is the applied force to the cart, $g = 9.8$ is the acceleration due to gravity, $l = 0.3$ is the length of the rod, $m = 0.2$ is the mass of the system to be balanced, $M = 0.5$ is the mass of the cart, $b = 0.1$ is the coefficient of friction of the cart, and $I = 0.006$ is the inertia of the pendulum. Using standard results in control theory [Kai80], one obtains the following state space realization for the inverted pendulum:

$$\begin{cases} \begin{bmatrix} \dot{\xi}_1 \\ \dot{\xi}_2 \\ \dot{\xi}_3 \end{bmatrix} = \begin{bmatrix} -0.1818 & 3.8977 & 0.5568 \\ 8.000 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} v, \\ \phi = [0 \ 0.5682 \ 1] \begin{bmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \end{bmatrix}. \end{cases}$$

Our objective is to design PID gains K_P , K_I , and K_D minimizing the cost function (5.7) with weighting factors $w_1 = w_2 = w_3 = 1$ and the closed loop system has a settling time (t_s) of less than 5 seconds and pendulum should not move more than 0.05 radians away from the vertical axis. The latter two constraints are treated the same as the stability constraint in Subsection 4.2 by penalizing the cost function (5.7). The synthesized gains are $K_P = 109.032$, $K_I = 1.2268$, and $K_D = 13.9945$. The closed loop system has $PM = +\infty$, $GM = 26237$,

$\gamma(b(e_{q1}) + b(e_{q2})) = 4.1705 \times 10^{-4}$, settling time $t_s = 0.4790$, and pendulum does not move more than 0.0098 radians away from the vertical axis.

7. CONCLUSION

We have presented a generic methodology to search for optimal controller implementations that minimize implementation errors in addition to traditional controller performance criteria. While we have instantiated the methodology using the LQR and LQG costs and quantization errors, our algorithm is more generally applicable to other performance criteria and other sources of modeling or implementation error. The controller synthesis algorithm can be seamlessly added to any design and automatic code generation tool flow to enhance its capability to generate correct-by-construction high performance controller software. By automatically synthesizing the minimal error controller, we sidestep the need for post-design verification.

REFERENCES

- [AM08] K. J. Astrom and R. M. Murray. *Feedback systems*. Princeton University Press, 2008.
- [AMST10] A. Anta, R. Majumdar, I. Saha, and P. Tabuada. Automatic verification of control system implementations. *In proceedings of EMSOFT*, pages 9–18, October 2010.
- [BR05] P. Belanovic and M. Rupp. Automated Floating-point to Fixed-point Conversion with the Fixify Environment. *In Proc. International Workshop on Rapid System Prototyping*, pages 172–178, 2005.
- [CF95] T. Chen and B. A. Francis. *Optimal sampled-data control systems*. Springer-Verlag, New York, 1995.
- [cmu] Control tutorial for matlab and simulink. Available online at <http://www.library.cmu.edu/ctms/ctms/>.
- [DM11] P. S. Duggirala and S. Mitra. Abstraction-refinement for stability. *in Proceedings of ICCPS*, pages 22–31, April 2011.
- [EKG12] S. Ebbesen, P. Kiwitz, and L. Guzzella. A generic particle swarm optimization function for Matlab. *American Control Conference (to appear)*, June 2012.
- [Fer10] Eric Feron. From control systems to control software. *IEEE Control Systems Magazine*, 30(6):50–71, December 2010.
- [GL94] M. Green and D. J. N. Limebeer. *Linear robust control*. Prentice Hall, August 1994.
- [Hes09] J. P. Hespanha. *Linear systems theory*. Princeton University Press, September 2009.
- [JLY07] M. Jiang, Y. P. Luo, and S. Y. Yang. Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm. *Information Processing Letters*, 102(1):8–16, April 2007.
- [Kai80] T. Kailath. *Linear systems*. Prentice-Hall, Inc., 1980.
- [Kal56] R. E. Kalman. Nonlinear aspects of sampled-data control systems. *in Proceedings of the Symposium on Nonlinear Circuit Analysis*, edited by J. Fox, Polytechnic Institute of Brooklyn, pages 273–313, 1956.
- [KE95] J. Kennedy and R. Eberhart. Particle swarm optimization. *In Proceedings of IEEE International Conference on Neural Networks*, pages 1942–1948, 1995.
- [LAS09] H. Liu, A. Abraham, and V. Snasel. Convergence analysis of swarm algorithm. *World congress on Nature and Biologically Inspired Computing*, pages 1714–1719, December 2009.
- [LCNT07] J. A. López, C. Carreras, and O. Nieto-Taladriz. Improved Interval-based Characterization of Fixed-point LTI Systems with Feedback Loops. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 26(11):1923–1932, 2007.
- [LGC⁺06] D. Lee, A. A. Gaffar, R. C. C. Cheung, O. Mencer, W. Luk, and G. A. Constantinides. Accuracy-Guaranteed Bit-width Optimization. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 25(10):1990–2000, 2006.
- [lps] lp_solve, a Mixed Integer Linear Programming (MILP) solver. Available online at <http://lpsolve.sourceforge.net/>.
- [LSG92] K. Liu, R. E. Skelton, and K. Grigoriadis. Optimal controllers for finite wordlength implementation. *IEEE Transactions on Automatic Control*, 37(9):1294–1304, September 1992.
- [LSW96] Y. Lin, E. D. Sontag, and Y. Wang. A smooth converse lyapunov theorem for robust stability. *SIAM Journal on Control and Optimization*, 34(1):124–160, 1996.
- [Moo66] R. Moore. *Interval Analysis*. Prentice Hall, 1966.
- [MSBZ07] A. Mallik, D. Sinha, P. Banerjee, and H. Zhou. Low-Power Optimization by Smart Bit-width Allocation in a SystemC-based ASIC Design Environment. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 26(3):447–455, 2007.
- [OCC⁺07] W. G. Osborne, R. C. C. Cheung, J. G. F. Coutinho, W. Luk, and O. Mencer. Automatic Accuracy-Guaranteed Bit-width Optimization for Fixed and Floating-point Systems. *In Proc. FPL*, pages 617–620, 2007.
- [PW06] A. Podelski and S. Wagner. Model checking of hybrid systems: from reachability towards stability. *in Proceedings of HSCC*, pages 507–521, April 2006.
- [PW07] Podelski and Wagner. Region stability proofs for hybrid systems. *in Proceedings of FORMATS*, pages 320–335, 2007.
- [SF97] J. Stolfi and L. H. Figueiredo. Self-validated Numerical Methods and Applications. *In Monograph for 21st Brazilian Mathematics Colloquium, Rio de Janeiro: IMPA*, 1997.
- [Wil85] D. Williamson. Finite wordlength design of digital Kalman filters for state estimation. *IEEE Transactions on Automatic Control*, 30(10):930–939, October 1985.

- [Wil89] D. Williamson. Optimal finite wordlength linear quadratic regulation. *IEEE Transactions on Automatic Control*, 34(12):1218–1228, December 1989.
- [Win93] G. Winskel. *The formal semantics of programming languages: an introduction*. MIT Press, February 1993.
- [ZKGS07] M. Zamani, M. Karimi-Ghartemani, and N. Sadati. FOPID controller design for robust performance using particle swarm optimization. *Journal of Fractional Calculus & Applied Analysis (FCAA)*, 10(2):169–188, 2007.
- [ZKGSP09] M. Zamani, M. Karimi-Ghartemani, N. Sadati, and M. Parniani. Design of a fractional order PID controller for an AVR using particle swarm optimization. *Control Engineering Practice*, 17(12):1380–1387, December 2009.
- [ZSKG09] M. Zamani, N. Sadati, and M. Karimi-Ghartemani. Design of an H_∞ PID controller using particle swarm optimization. *International Journal of Control, Automation, and Systems (IJCAS)*, 7(2):273–280, April 2009.

¹MAX PLANCK INSTITUTE FOR SOFTWARE SYSTEMS, KAISERSLAUTERN, GERMANY

E-mail address: `rupak@mpi-sws.org`

URL: `http://www.cs.ucla.edu/~rupak`

²DEPARTMENT OF COMPUTER SCIENCE, UNIVERSITY OF CALIFORNIA AT LOS ANGELES, LOS ANGELES, CA 90095

E-mail address: `indranil@cs.ucla.edu`

URL: `http://www.cs.ucla.edu/~indranil`

³DEPARTMENT OF ELECTRICAL ENGINEERING, UNIVERSITY OF CALIFORNIA AT LOS ANGELES, LOS ANGELES, CA 90095

E-mail address: `zamani@ee.ucla.edu`

URL: `http://www.ee.ucla.edu/~zamani`